Designer Adaptation in Adaptive Hypermedia Authoring

Alexandra I. Cristea and Arnout de Mooij
Faculty of Mathematics and Computing Science, Information Systems Department
Technische Universiteit Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
a.i.cristea@tue.nl

Abstract

Recently, the importance of creating authoring support for adaptive hypermedia system design offering multi-modality and personalization is becoming evident [4][5][6][7][1]. In the process of designing such authoring support, we discovered that given the difficulty of adaptive hypermedia authoring, i.e., in designing different levels of abstraction, alternatives, multiple links, etc., it would be beneficial to also attend to the authoring needs and adapt to the author. Therefore, this paper describes for the first time an attempt of adaptation not only to the student, but also to the designer.

1. Introduction

1.1 Design of Adaptive hypermedia

A hypermedia system is a database-like software system, which can be accessed using a hypermedia tool, as for instance the web via a web browser [1]. A hypermedia system is said to be adaptive if it can automatically adapt to goals, needs or, e.g., new conditions, which can be deduced, for example, from actions the system user undertakes. For instance, in adaptive courseware the adaptation is reflected in the various ways and orders in which the study material is presented to the different students. Basically, the more alternatives there are, the higher the adaptation degree. This flexibility is beneficial as long as it serves a specific learning goal, but can also lead to psychological pressure and unwanted effects. To find the exact balance between flexibility versus stability and predictability is extremely important and a challenging research in itself, but we are not going to go into details about this in this paper.

1.2. From MyEnglishTeacher to MOT

To study various types of adaptation, as well as methods for their efficient design, we use an adaptive course system design tool. The present system extends a previous tool for constructing adaptive hypermedia systems, My English Teacher (MyET [8][12]) and its successor My Online Teacher (MOT [13]). Although there are many tools for course development, there are only few other examples of tools trying to create adaptive courseware, as shown in [3]. MyET allowed teachers to create concepts and concept maps to model their courses. Based on these concept maps the teacher should be able to construct lessons, to be presented to the student in an adaptive way. The part of (independent) lesson construction was not yet available in MyET and the way in which concept maps could be created was too restricted. Moreover, MyET used files to store all the information, while a database is preferred. The goal was to extend MyET into 'My Online Teacher' (MOT v.2), for a more general audience. This is why the first implementation testing is done via a 'Neural Networks' course for third year students in Computer Science. In design terms, the issue was to make it possible to construct complete concept maps and lessons, stored in a database, and to demonstrate some simple features of automatically binding concepts for adaptation purposes.

2. Goals

2.1. Initial goals

An adaptive (web) hypermedia course is a hypermedia system that can be used by a student to learn about a certain subject via, e.g., a web browser. The basic feature of such a system is that it tries to interpret the students' current knowledge (and often, other student parameters and characteristics as well) in order to adapt itself to his learning needs. Ideally, there is no need for a human teacher. The student can perform actions such as choosing topics he wants to learn about, asking for more information or solving exercises. Depending on the actions the student takes (for example the pages visited, or the results of exercises) the course transparently adapts to the student's needs.

We have delimited the main steps of the creation of an *adaptive lesson design system* as being the creation of:

1. A tool for manipulating concept maps.
2. A method for calculating correspondence weights between concept attributes.
3. A tool for constructing lessons based on a concept map.

2.2. Designer Adaptation

Adaptation to the teacher is similar to adaptation to the student, if we consider that adaptation occurs with regard to a goal (be it a learning, or a design goal) or if it comes as a response to a need [9]. However, this new type of adaptation is quite different, if we consider the typical adaptation to (student) user preferences. Such (designer) user preferences adaptation is not taken into consideration here. The paper shows the design and implementation of this new variant of adaptivity, within the larger context of designing an authoring support system for adaptive hypermedia. The adaptation of a course under design to the design goal can take many forms. This first version illustrates (semi-) automatic course linking. Thereby, we can claim to lay the basis for a course that builds (or writes) itself.

3. Database design

The database was to be implemented according to the ER-diagram depicted in Fig. 1, representing the initial static UML classes, and which can be divided into two parts: the *concept domain*, formed by the left side of the diagram, and the *course* (or lesson hierarchy) on the right side of the diagram. These two parts are connected by

3.1. Concept domain

A concept contains one or more sub-concepts, which are concepts on their turn, hence inducing a hierarchical (tree) structure of concepts.

Each concept is a set of concept attributes. These attributes hold pieces of information about the concept they belong to. Several kinds of attributes are possible, corresponding to the different attribute instances in the diagram. For example, a concept can have a 'title'-attribute, a 'description'-attribute or 'example'-attribute.

Concept attributes can be related to each other. Such a relation, characterized by a label and a weight, indicates that their contents treat similar topics.

Exercises are modeled as special concepts, because they have their own hierarchical structure within the greater concept structure, while they actually belong to one (non-exercise) concept.

3.2. Course

A lesson contains sub-lessons, which are lessons on their turn, hence creating a hierarchical structure of lessons. Sub-lessons within a lesson can be OR-connected (being lesson alternatives) or AND-connected. To facilitate this, a lesson contains a lesson attribute (L-

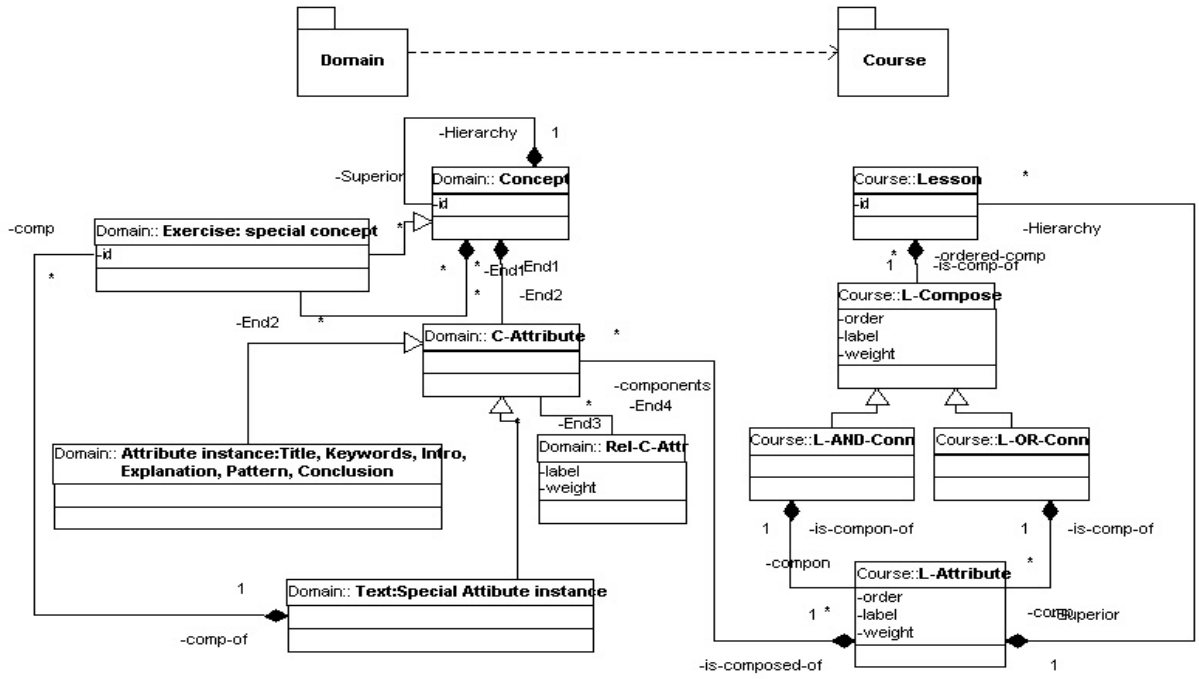

Figure 1. Initial ER-diagram

means of the relation between the C-Attribute (concept attribute) and the L-Attribute (lesson attribute).

Attribute in the diagram), which in its turn contains a holder for OR-connected sub-lessons (L-OR-Conn) or a holder for AND-connected sub-lessons (L-AND-Conn).

The holder contains the actual sub-lessons in a specified order.

A lesson attribute contains, besides the sub-lesson holders, one or more concept attributes. This is the link with the concept domain. The idea is that the lesson puts pieces of information that are stored in the concept attributes together in a suitable way for presentation to a student.

In the database implementation phase and even in the later phase of system implementation and adding the feature of calculating relatedness relations between concept attributes some changes were made to the ER-diagram, but this will not be detailed in the current paper.

4. Calculating relatedness relations

The assignment describes so called 'relatedness relations' between concept attributes. Concept attributes are related when they share a common topic. It turned out to be more logical to record this relation type at concept level, so that a 'relatedness relations' marks the existence of a relation between concepts. If the relatedness is induced from an attribute level, we took the design decision to keep the name of that attribute as the semantic label for the respective relatedness relation.

The system was required to help the teacher in determining the relatedness relations by calculating correspondence weights between pairs of concepts. There are several ways of computing such links, some symbolic, some sub-symbolic. For a simplified illustration of (semi-)automatic binding, we took the design decision to base these correspondence weights on the number of occurrences of the keywords of one concept in the attribute contents of the other concept.

4.1. The initial plan for relatedness calculation

We consider the concept map of the courseware to be determined by the tuple $<C,L>$, where C represents the set of concepts and L the set of links, and a concept $c \in C$ is defined by its set of attributes, A_c (where $A_c \supseteq A_{min}$; A_{min} is the minimal set of attributes required for each concept to have[1]). As all the sets above are finite, they can be given (relative) identification numbers. Therefore, concept c is determined (and therefore can be referred to) by its identification $i \in \{1,\ldots,C\}$ (where C=card(C)) and the attributes of concept i are $a_i[h]$, with $h \in \{1,\ldots,A_i\}$ and $A_i \geq A_{min}$ (where A_i=card(A_c) and A_{min}=card(A_{min})). Moreover, a special attribute of each concept, $a_i[2]=\{(k_i[s])|\ s= 1,\ldots,K_i\}$, is the list of keywords for concept i (with K_i the number of keywords of attribute $a_i[2]$ called 'keyword' of concept i). This 'keyword' attribute is obligatory, therefore $a_i[2] \in A_{min}$.

With the above notations, we can express the number of occurrences of keyword $k_i[s]$, with $s \in \{1,\ldots,K_i\}$ of concept $i \in \{1,\ldots,C\}$ in an attribute $a_j[h]$, with $h \in \{1,\ldots,A_j\}$ of concept $j \in \{1,\ldots,C\}$ as being given by $occ_{ij}(k_i[s], a_j[h])$. If:

$$maxocc_{ij}(k_i[s], a_j[h]) = count(\text{words in } a_j[h]);$$

is the maximum possible number of occurrences of $k_i[s]$ in $a_j[h]$, then $occ_{ij}(k_i[s], a_j[h]) / maxocc_{ij}(k_i[s], a_j[h]) \in [0, 1]$.

When we add these values over all keywords of i and all attributes of j and divide the result by the number of keywords of i and by the number of attributes of j we get a value which is also between 0 and 1, indicating the level of correspondence between concept i and concept j.

Therefore, for $\forall$concepts $i, j \in \{1,\ldots,C\}$ we can define:

$$correspondence\_directed(i,j) = \frac{\sum_{h=1}^{Aj}\sum_{s=1}^{Ki}\frac{occ_{ij}(k_i[s],a_j[h])}{maxocc_{ij}(k_i[s],a_j[h])}}{K_i * A_j}$$

As the name says, this number only looks at one direction of the relation between concepts i and j. If we consider also the reverse direction, a better measure for the correspondence between the concepts can also contain a weighted reverse correspondence, as follows:

$$correspondence(i,j) = \alpha * correspondence\_directed(i,j) + \beta * correspondence\_directed(j,i)$$

with $\alpha, \beta \in [0,1]$ and $\alpha+\beta=1$

If $\alpha=\beta=0,5$ the following relation also holds:

$$correspondence(i,j) = correspondence(j,i)$$

To further fine-tune the relatedness calculation an importance weight can be assigned to each type of attribute to be able to give certain attributes (e.g., the title, $a_i[1]$, or the keywords, $a_i[2]$, attribute) a stronger influence on the correspondence weight. Therefore, if we consider $importance(a_j[h]) \in [0,1]$ to be the value of the importance (or weight) of attribute $a_j[h]$ and if:

$$\Sigma_{h=1,\ldots,Aj}\ (importance(a_j[h]) = 1;$$

The formula for *correspondence_directed* becomes:

$$correspondence\_directed(i,j) = \frac{\sum_{h=1}^{Aj}\sum_{s=1}^{Ki}\frac{importance(a_j[h]) * occ_{ij}(k_i[s],a_j[h])}{maxocc_{ij}(k_i[s],a_j[h])}}{K_i * A_j}$$

A more generalized formula is:

$$correspondence\_directed(i,j) = \frac{\sum_{h=1}^{Aj}\sum_{s=1}^{Ki}\frac{importance(a_j[h]) * occ_{ij}(k_i[s],a_j[h])}{maxocc_{ij}(k_i[s],a_j[h])}}{K_i * \sum_{h=1}^{Aj} importance(a_j[h])}$$

[1] by the adaptive course design constrains, that aim at creating concepts annotated with sufficient meta-data

The definition of *correspondence(i,j)* remains unchanged. The system will then calculate *correspondence(i,j)* for each pair of concepts *i* and *j* and suggest a relatedness relation between the two concepts when this value exceeds a certain threshold. The teacher can choose to add this relatedness relation or to ignore it. When the teacher decides to add a relatedness relation he can give it a label and a weight (by default the correspondence weight, *correspondence_directed(i,j)*, is proposed).

4.2. A second plan for relatedness calculation

Relatedness relations will have to have a type. Using the formula as described above, it is impossible to see in what way two concepts are related, because the correspondence weights of all keywords and attributes are used to get the correspondence weight.

A better idea is therefore to calculate correspondence weights per attribute as follows.

With the above notations, for ∀ concepts *i*, *j* and ∀ attributes a_j[h] of *j*:

$$correspondence\_directed(i, j, a_j[h]) = \frac{\sum_{s=1}^{Ki} \frac{occ_{ij}(k_i[s], a_j[h])}{maxocc_{ij}(k_i[s], a_j[h])}}{K_i}$$

The *attribute type* of attribute a can be used as a label (or even type) of the relation, with the weight of the relation given by *correspondence_directed(i, j, a_j*[h]*)*.

For example, concepts A and B can have a relatedness relation of type 'title', with the weight given by *correspondence_directed(A,B,a_B*[1]*)* and also a relatedess relation of type 'text', with the weight given by *correspondence_directed(A, B, text attribute of B)*. In this way, more than one relatedness relation between two concepts can exist. However these relations will have different types, indicating in what way the concepts are related.

Optionally, a value:

$$correspondence(i,j,a_j[h]) = (correspondence\_directed(i,j,a_j[h]) + correspondence\_directed(j,i,a_i[h'])) / 2$$

can be used for all concepts *i*, *j* and all attributes $a_j[h]$ of *i* and all attributes $a_i[h']$ of *j*, $a_j[h]$ and $a_i[h']$ having the same type (i.e., *h=h'*). Please note that concepts may not have attributes of the same type (if $a_j[h] \in \mathbb{A}_{min}$, then the existence of an attribute of the same type, $a_i[h'] \in \mathbb{A}_{min}$, given the condition *h=h'* is fulfilled, is guaranteed; if $a_j[h] \in \mathbb{A}_c - \mathbb{A}_{min}$, such guarantee does not exist, and it is up to the course designer to ensure the existence of recognizable types, if desired).

5. Relatedness Computations Implementation

Figure 2 shows a screenshot of the list of possible connections the system automatically finds and their suggested weights for concept 'Theorem of Batch Perceptron Convergence' from a Neural Networks course.

Current concept: Theorem of Batch Perceptron Convergence

The following relatedness weights are found:

Concept name	Relation Type	Relation weight
NN Introduction [add]	text	0.217
Learning [add]	text	0.245
DISCRETE-NEURON PERCEPTRONS [add]	keywords	0.515
DISCRETE-NEURON PERCEPTRONS [add]	text	0.308
DISCRETE-NEURON PERCEPTRONS [add]	title	0.539
Introduction [add]	keywords	0.471
Introduction [add]	text	0.394
Introduction [add]	title	0.204
One-layer Discrete Perceptrons [add]	keywords	0.354
One-layer Discrete Perceptrons [add]	title	0.289
NN Introduction [add]	text	0.217
Learning in the ANN [add]	text	0.245
Discrete -Neuron Perceptrons [add]	keywords	0.515
Discrete -Neuron Perceptrons [add]	text	0.308

Figure 2. Automatic relatedness relations

The course designer (teacher) can accept or reject them, as well as change weights or labels (Fig. 3). This screen appears after pressing 'add' in the previous one.

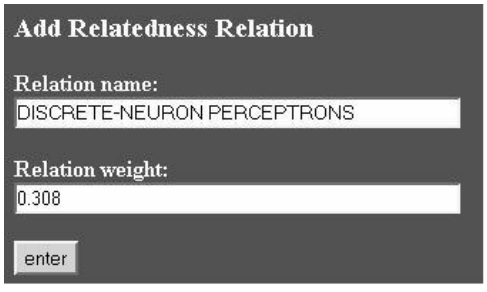

Figure 3. Adding relatedness relations

6. Project evaluation

6.1 Planning

It turned out that lesson modeling (based on concept maps) was more difficult then expected. At first, concept maps and lessons seemed to be much alike, so we expected to use the same database structure for both. However, this proved to be a bad idea and we had to make some great extensions to the database.

Also, it became apparent that not all desired functionality of the user interface was stated in the URD. Especially the calculation of the relatedness relations turned out to be somewhat more complicated then expected. Even now, this part is not really fulfilled satisfactory. Furthermore, the first versions of the interface were not very enjoyable to use, so changes had to be made a couple of times. For example, HTML-lists used for displaying lessons and concept maps had to be replaced by collapsible lists written in JavaScript, etc.

6.2 Evaluation of the system

The delivered system satisfies in principle all user requirements and is efficient in it's use. However, some critical remarks can be made.

It is a minimal system with no extra's and some open ends. For example, the interface is non-graphical, while the representation of concept maps could possibly benefit a lot from using graphical elements. Some other features that are lacking are warnings (e.g. when removing concepts), error messages (when performing illegal actions) and security issues (preventing users from viewing or changing other users' concept maps or lessons). Finally, much can be done to improve the calculation and typing of relatedness relations. All these provide some work for the future.

7. Conclusions

In this paper we have presented a system for designing adaptive hypermedia, instantiated within an educational setting. We have briefly shown the design and implementation steps of this system. We have argued from the start that the authors of adaptive hypermedia have a considerably difficult task, compared to authors of regular hypermedia, for example. Based on this assumption, we have endeavored to construct a support system that adapts to the design goal. Therefore, the focus was on the adaptivity to the designer. This is an extended use of this term, if compared to the user adaptivity we are used to.

Adaptation to the designer, the way we see it, can be various: link -, content adaptation, hints, different design levels, templates suggesting, but also recognition, collaboration support [6]. However, in this paper we have only shown a small demonstrative example of automatic linking. Moreover, as suggested in the evaluation section, there is space for improvement even in automatic linking: instead of deterministic functions, sub-symbolic clustering techniques can be used [10].

In this way we have made a step towards hypermedia that builds (or writes) itself.

8. Acknowledgements

This research is linked to the European Community Socrates Minerva project "Adaptivity and adaptability in ODL based on ICT" (project reference number 101144-CP-1-2002-NL-MINERVA-MPP).

9. References

[1] *ADAPT*: http://wwwis.win.tue.nl/~alex/HTML/Minerva/

[2] 2L690: Hypermedia Structures and Systems, Lecturer: Prof. De Bra, http://wwwis.win.tue.nl/~debra/2L690/

[3] Brusilovsky, P.: Adaptive hypermedia, *User Modeling and User Adapted Interaction*, Ten Year Anniversary Issue (Alfred Kobsa, ed.) 11 (1/2), 2002, 87-110.

[4] Calvi, L. and Cristea, A.I.: Towards Generic Adaptive Systems Analysis of a Case Study, AH 2002, *Adaptive Hypermedia and Adaptive Web-Based Systems*, LNCS 2347, Springer, 79-89.

[5] Cristea, A.I. and De Bra, P.: Towards Adaptable and Adaptive ODL Environments, *E-Learn'02*, Montreal, Canada, AACE, October 2002, pp. 232-239.

[6] Cristea, A.I., Okamoto, T. and Kayama, M.: Considerations for Building a Common Platform for Cooperative &Collaborative Authoring Environments, *E-Learn'02*, AACE, October 2002, pp. 224-231.

[7] Cristea, A.I. and Aroyo, L.: Adaptive Authoring of Adaptive Educational Hypermedia, AH 2002, *Adaptive Hypermedia and Adaptive Web-Based Systems*, LNCS 2347, Springer, 122-132.

[8] Cristea, A.I., Okamoto, T. and Belkada, S.: Concept Mapping for Subject Linking in a WWW Authoring Tool: MyEnglishTeacher: Teachers' Site, ANNIE'00, ASME.

[9] *IMS* (Instruct. Manag. System): http://www.imsproject.org

[10] Kayama, M., Okamoto, T. and Cristea, A.I.: Exploratory Activity Support Based on a Semantic Feature Map, *AH'00*, LNCS 1892, Springer, 347-350.

[11] Loeber, S. and Cristea, A.I. (2002), A WWW Information-Seeking Process Model, *ISSEI 2002*, CBMO workshop, 22 - 27 July, 2002, England.

[12] *MyET*:http://wwwis.win.tue.nl/~alex/MyEnglishTeacher/TeachersSite/index.html

[13] *MOT*:http://wwwis.win.tue.nl/~alex/MOT01/TeachersSite-html/index.html

[14] Wu, H., Houben,G.-J., De Bra, P.: AHAM: A Reference Model to Support Adaptive Hypermedia Authoring, Informatiewetenschap 1998, Ed. E. De Smet, Antwerp, Belgium, 11 December 1998, 51-76.